rdcoils

Initialize the coils data with Fourier series.

[called by: focus.] [calls: .]

0.1 overview

1. If `case_coils=1`, then the Fourier series will be used for represent the coils.
2. The basic equations about the Fourier representation is,

$$x = X_{c,0} + \sum_{n=1}^{N} \left[X_{c,n} \cos(nt) + X_{s,n} \sin(nt) \right], \tag{1}$$

$$y = Y_{c,0} + \sum_{n=1}^{N} \left[Y_{c,n} \cos(nt) + Y_{s,n} \sin(nt) \right], \tag{2}$$

$$z = Z_{c,0} + \sum_{n=1}^{N} \left[Z_{c,n} \cos(nt) + Z_{s,n} \sin(nt) \right], \tag{3}$$

0.2 Initilization

There are several ways to initialize the coils data.

1. `case_init = 0` : Toroidally placing `Ncoils` circular coils with a radius of `init_radius` and current of `init_current`. The ith coil is placed at $\zeta = \frac{i-1}{Ncoils}\frac{2\pi}{Nfp}$.
2. `case_init = 1` : Read coils data from **.ext.coil.xxx** files. xxx can vary from 001 to 999. Each file has such a format. This is the most flexible way, and each coil can be different.

```
#type of coils; name
    1 "Module 1"
# Nseg I Ic L Lc Lo
   128 1.0E+07 0 6.28 1 3.14
# NFcoil
1
# Fourier harmonics for coils ( xc; xs; yc; ys; zc; zs)
3.00 0.30
0.00 0.00
0.00 0.00
0.00 0.00
0.00 0.00
0.00 0.30
```

3. `case_init = -1` : Get coils data from a standard coils.ext file and then Fourier decomposed (normal Fourier tansformation and truncated with $NFcoil$ harmonics)

0.3 Discretization

1. Discretizing the coils data involves massive triangular functions in nested loops. As shown in Eq.(**??**), the outside loop is for different discrete points and for each point, a loop is needed to get the summation of the harmonics.
2. To avoid calling triangular functions every operations, it's a btter idea to allocate the public triangular arrays.

$$cmt(iD, iN) = \cos(iN \frac{iD}{D_i} 2\pi); iD = 0, coil(icoil)\%D; \ iN = 0, coil(icoil)\%N \tag{4}$$

$$smt(iD, iN) = \sin(iN \frac{iD}{D_i} 2\pi); iD = 0, coil(icoil)\%D; \ iN = 0, coil(icoil)\%N \tag{5}$$

3. Using the concept of vectorization, we can also finish this just through matrix operations. This is in **fouriermatrix**.

```
subroutine fouriermatrix(xc, xs, xx, NF, ND)
nn(0:NF, 1:1) : matrix for N; iN
tt(1:1, 0:ND) : matrix for angle; iD/ND*2pi
nt(0:NF,0:ND) : grid for nt; nt = matmul(nn, tt)
xc(1:1, 0:NF) : cosin harmonics;
xs(1:1, 0:NF) : sin harmonics;
xx(1:1, 0:ND) : returned disrecte points;
```

```
xx = xc * cos(nt) + xs * sin(nt)
```

4. Actually, in real tests, the new method is not so fast. And parallelizations are actually slowing the speed, both for the normal and vectorized method.